1,关联查询(理解)

- 目的: 知道书籍查英雄, 知道英雄能够查书籍
- 格式:
    - 1, 知道书籍条件查英雄
        - HeroInfo.objects.filter(hbook__字段=值)
    - 2, 知道英雄能够查书籍
        - BookInfo.objects.filter(heroinfo__字段=值)

2,related_name

- 目的: 知道了一方, 能够快速的查询出另外一方的数据
- 格式:
    - 1, *知道了郭靖,查询郭靖属于那本书籍* (多查一)
        - *hero = HeroInfo.objects.get(hname="郭靖")*
        - hero.hbook
    - 2, *知道了天龙八部,查询天龙八部有哪些英雄*(一查多)
        - book = BookInfo.objects.get(btitle="**天龙八部**")
        - book.heroinfo_set.all()
- 注意点:
    - 在一查多的时候, 如果不想使用系统提供的关系属性, 可以使用related_name指定

```
hbook = models.ForeignKey(
  BookInfo, on_delete=models.CASCADE, related_name="heros", verbose_name='图书')  # 外键
```

        - book = BookInfo.objects.get(btitle="**天龙八部**")
        - book.heros.all()

3,数据库修改,删除(掌握)

- 目的: 能够使用orm提供的方法,对数据进行修改和删除
- 操作方法:

    - 1, 修改: update()
    - 2, 删除: delete()

4,惰性,缓存,限制(理解)

- 目的: 知道惰性,缓存,限制,含义
    - 惰性: 只有用到了才去查询

```
books = BookInfo.objects.all()
[book.btitle for book in books] #才转换成sql语句
```

    - 缓存: 使用了前面查询的结果, 没有做新的查询

```
[book.btitle for book in books] #才转换成sql语句
```

    - 限制: 使用切片的方式得到想要的结果

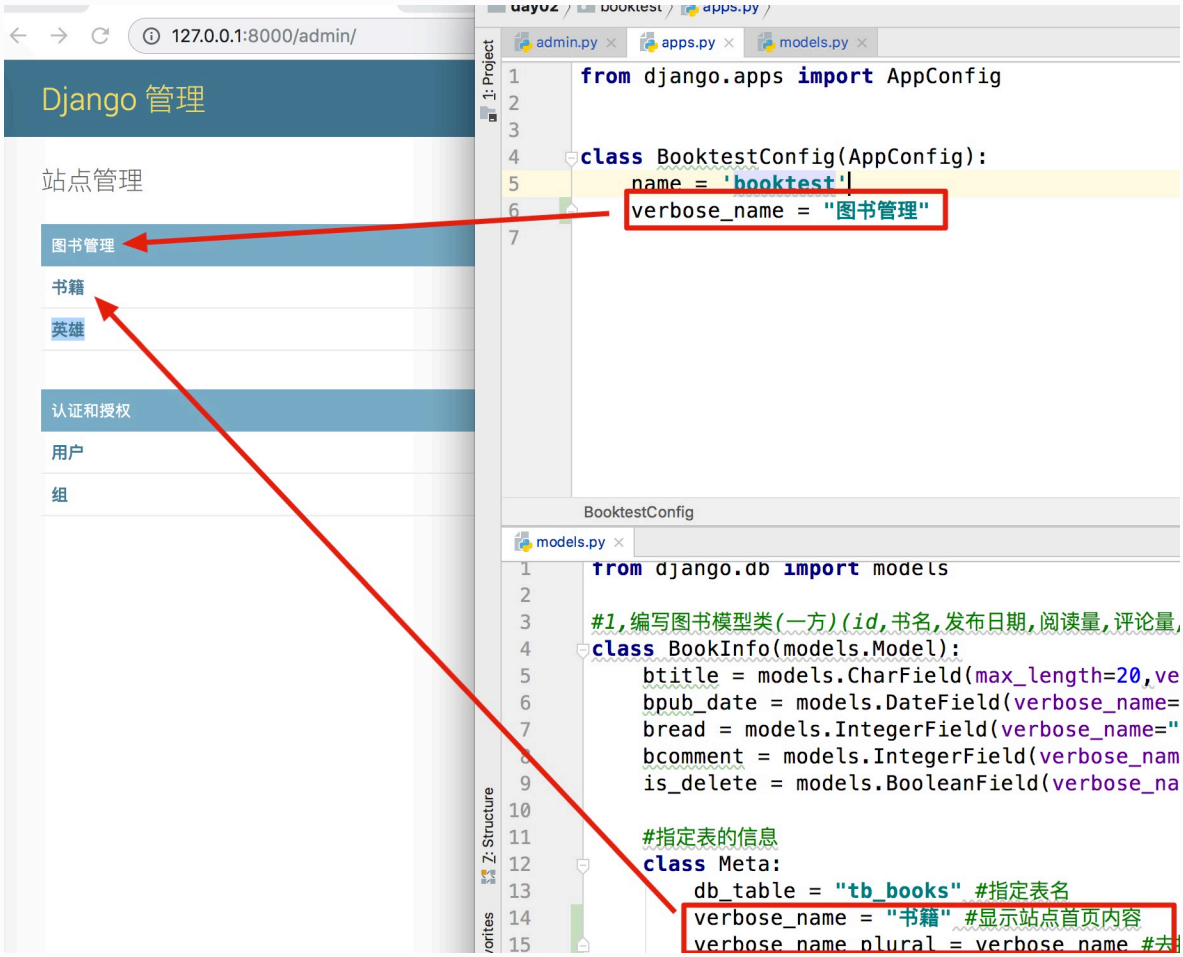
```
books = BookInfo.objects.all()[0:2]
```

5,创建管理员账号(掌握)

- 目的: 能够通过django中提供的命令来创建管理员的账号和密码
- 创建命令
    - 1, python manage.py createsuperuser
    - 2, 根据提示输入, 账号,邮箱,密码;
    - 3,输入127.0.0.1:8000/admin/login

6,模型类注册(掌握)

- 目的: 知道如何将模型类注册到站点中, 并配置首页的显示



7,定义Admin管理类(掌握)

- 目的: 能够定义模型类的, 站点管理类,来配置模型类对应界面的显示
- 配置过程:
  - 1, 定义管理类 booktest/admin.py

```
class BookInfoAdmin(admin.ModelAdmin):
    pass
```

  - 2, 配置字段

```python
class BookInfoAdmin(admin.ModelAdmin):
    #1,显示条数
    list_per_page = 10

    #2,动作栏
    actions_on_top = False
    actions_on_bottom = True

    #3,显示的字段
    list_display = ("id","btitle","bread","bcomment")
    pass
```

- 3,将管理类注册到admin.site中

```python
admin.site.register(BookInfo,BookInfoAdmin)
```

8, 方法作为字段显示

- 目的: 能够配置方法在站点管理中显示
- 格式:
  - 1, 在模型类中定义方法

```python
class BookInfo(models.Model):
    ...
    #获取日期
    def my_date(self):
        return self.bpub_date
    my_date.short_description = '我的日期'
    my_date.admin_order_field = "bpub_date"
```

  - 2, 添加到管理类的list_display中
    - list_display = ("**id**","**btitle**","**bread**","**bcomment**","**my_date**")

8,过滤&搜索配置(理解)

- 目的: 能够配置搜索, 过滤的显示

- 显示配置

```
class BookInfoAdmin(admin.ModelAdmin):
    ...
    #4,过滤
    list_filter = ("btitle","bpub_date")

    #5,搜索
    search_fields = ("btitle",)
```

9, 关联对象

- 目的: 显示关联对象的数据
- 格式:
  - 1, 在书籍中显示英雄

```
models.py
class BookInfo(models.Model):
    ...
    #配置英雄显示
    def my_heros(self):
        # return self.heros.all()
        return [hero.hname for hero in self.heros.all()]

    my_heros.short_description = "英雄"


admin.py
list_display = ("id","btitle","bread","bcomment","my_date","my_heros")
```

  - 2,在英雄中显示书籍

```python
class HeroInfo(models.Model):
    ...
    hbook = models.ForeignKey(
      BookInfo, on_delete=models.CASCADE, related_name="heros", verbose_name='图书')  # 外键

    #显示书籍阅读量
    def book_bread(self):
        return self.hbook.bread

list_display = ("id","hname","hbook","book_bread")
```

10,编辑页面(理解)

- 目的: 编辑页面的展示修改
- 配置

```python
class BookInfoAdmin(admin.ModelAdmin):
    ...
    #6,编辑字段
    # fields = ("btitle","bpub_date")

    #7,fieldsets以组的形式表示编辑字段(和上面的fields互斥)
    fieldsets = (
        ("基础组", {
            'fields': ('btitle', 'bpub_date')
        }),
        ('高级组', {
            'classes': ('collapse',), #折叠
            'fields': ('bread', 'bcomment',"is_delete"),
        }),
    )
```

11, 编辑页关联对象(理解)

- 目的: 关联对象的展示, 可以配置出两种显示形式(横向列表, 纵向列表)
- 配置过程:
    - 1, 定义关联的编辑,模型类

```
class HeroInfoShow(TabularInline): #横向展示
    model = HeroInfo #关联的模型类
    extra = 0 #额外展示的内容
    pass

```

    - 2, 添加到管理类的inlines中

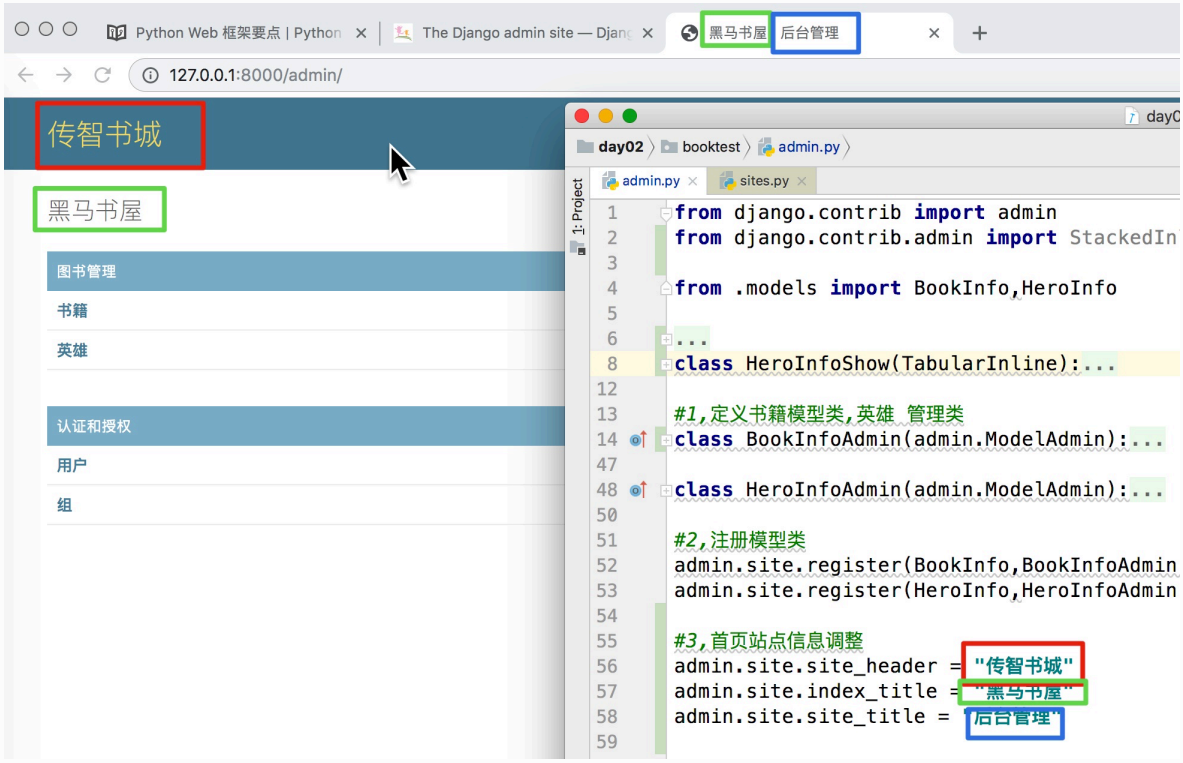
```

class BookInfoAdmin(admin.ModelAdmin):
    ...

    #8,关联模型类展示
    inlines = [HeroInfoShow]
```

12,站点信息调整(理解)

- 目的: 修改首页站点信息

- 

13,图片上传(理解)

- 目的: 能够给模型类添加图片字段, 知道如何上传图片
- 操作流程:
  - 1, 添加bimage字段,指定上传的文件夹

```
class BookInfo(models.Model):
    ...
    bimage = models.ImageField(verbose_name="图片",null=True,blank=True,upload_to="booktest")

```

  - 2, 迁移
    - python manage.py makemigrations
    - python manage.py migrate
  - 3,将字段添加到分组中

```python
class BookInfoAdmin(admin.ModelAdmin):
    ...
    #7,fieldsets以组的形式表示编辑字段(和上面的fields互斥)
    fieldsets = (
        ("基础组", {
            'fields': ('btitle', 'bpub_date',"bimage")
        }),
        ...
    )

```

    - 4,设置公共的图片存储的文件路径 settings.py
        - MEDIA_ROOT = os.path.join(BASE_DIR,"**static**")
    - 5,在页面上传即可

- 注点意:
    - 最终上传的文件目录是: 和根应用同级下面,生成 static/booktest

14,web应用模式(理解)

- 目的: 了解前后端分离不分离的形式
- 分离: 只需要返回json数据即可
- 不分离: 只有一个应用服务器, 既要查询数据, 又要渲染页面

15,restful介绍

- 目的:
    - 1, 写接口的时候以后要遵守restful规范, 好处:有利于团队开发
    - 2, RESTful是一种开发理念。REST是设计风格而不是标准。

16,restful需求分析

- 目的: 了解restfult10中设计风格
    - 常见的
        - 1, 请求方式规范: get,post,put,delete
        - 2, 请求路径: 不能出现动词
        - 3, 状态码

- 4, 域名部署
- 5, 超媒体
- 6, other, 返回数据的时候,返回json
- ……

17,查询所有数据

18,创建对象

19,获取单个对象

20,修改单个对象

21,删除单个对象